Multiway Partitioning with Pairwise Movement*

Jason Cong and Sung Kyu Lim
UCLA Department of Computer Science, Los Angeles, CA 90095
{cong,limsk}@cs.ucla.edu

Abstract

It is known to many researchers in the partitioning community that the recursive bipartitioning approach outperforms the direct non-recursive approach in solving the multiway partitioning problem. However, little progress has been made to identify and overcome the weakness of the direct (alternatively called flat) approach. In this paper, we make the first observation that the performance of iterative improvement-based flat multiway partitioner K-FM [10, 11] is not suitable for today's large scale circuits. Then, we propose a simple yet effective hill-climbing method called PM (Pairwise cell Movement) that overcomes the limitation of K-FM and provides partitioners the capability to explore wider range of solution space effectively while ensuring convergence to satisfying suboptimal solutions. The main idea is to reduce the multiway partitioning problem to sets of concurrent bipartitioning problems. Starting with an initial multiway partition of the netlist, we apply 2-way FM [7] to pairs of blocks so as to improve the quality of overall multiway partitioning solution. The pairing of blocks is based on the gain of the last pass, and the Pairwise cell Movement (PM) passes continue until no further gain can be obtained. We observe that PM passes are effective in distributing clusters evenly into multiple blocks to minimize the connections across the multiway cutlines. Our iterative improvement-based flat multiway partitioner K-PM/LR improves K-FM by a surprising average margin of up to 86.2% and outperforms its counterpart recursive FM by up to 17.3% when tested on MCNC and large scale ISPD98 benchmark circuits [1].

1 Introduction

The divide-and-conquer paradigm is regarded indispensable for solving today's complex VLSI layout problem; the problem must be partitioned into smaller subproblems until they are small enough to be solved effectively and efficiently. Circuit partitioning is a technique to divide the given circuit into a collection of subcircuits, which has been an active area of research for at least a quarter of a century. The main reason that partitioning plays a critical role in design task today is the enormous increase of system complexity along with substantial advances in VLSI system design and fabrication. Among many solutions devised to solve the partitioning problem, iterative improvement method such as FM [7] is accepted as de facto standard in handling today's large scale circuits due to (i) linear-time behavior (ii) flexibility in handling various constraints, (iii) controllable cutsize/runtime tradeoff.

Multiway circuit partitioning divides the given circuit into a predetermined number (> 2) of subcircuits. The standard objective is to minimize the number of nets among all partitions while satisfying various constraints such as lower and upper bounds on the area and pin count of each partition. Some of the previous works include recursive KL [9], generalization of FM [10, 11], primal-dual [12], spectral multiway ratio-cut [3], geometric embedding [2], multilevel-based [8], and dual net-based [4] method.

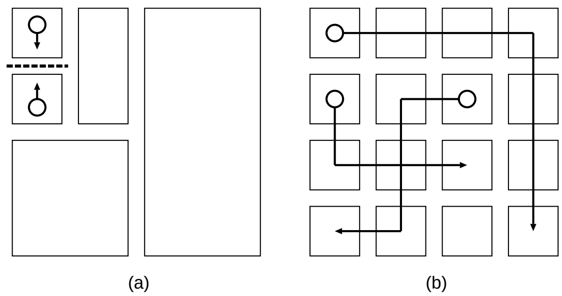

Figure 1: (a) local cell move under recursive approach, (b) global cell move under flat approach

There are two primary approaches for generating multiway partitioning solution; *recursive* or *flat*. The recursive approach applies bipartitioning recursively until the desired number of partitions is obtained, whereas the flat approach partitions the circuit directly. We note that cells move locally across the current level cutline in case of recursive bipartitioning as depicted in Figure 1-(a), whereas flat approach enables cells to move between any arbitrary blocks, promoting global change in the current configuration as depicted in Figure 1-(b). Although it has remained controversial, recursive approach is preferred in practice despite the lack of global information and its greedy nature. This is mainly due to its computational simplicity and cost-effectiveness. In addition, recent advances in enhancing iterative improvement bipartitioning algorithms have also made recursive approach more and more attractive in solving multiway partitioning. On the other hand, little progress has been made to improve iterative improvement-based flat multiway partitioning algorithms such as K-FM [10, 11] despite the potential gain from the availability of more global information and larger solution space. One major drawback of K-FM that is consensus among CAD researchers is that it is very susceptible of being trapped into a local minima that is far from being optimal. This is especially true for flat multiway partitioning where the partitioner can easily make wrong decision while dealing with more number of candidate cells and directions to move.

In this paper, we propose a simple yet effective approach to enhance the performance of K-FM by reducing the multiway partitioning problem to sets of concurrent bipartitioning problems. Starting with an initial K-way partition of

*This work was partially supported by NSF Young Investigator Award MIP-9357582, and Fujitsu Laboratories at America under the 1997-98 California MICRO Programs.

the netlist, we apply bipartitioning heuristic FM [7] to pairs of blocks so as to improve the quality of overall multiway partitioning solution. The pairing of blocks is based on the gain of the last pass, and the Pairwise cell Movement (PM) passes continue until no further gain can be obtained. This method is shown effective in optimizing standard cost 1 and cost $k-1$ (to be defined in Section 2) metrics. We adapt existing LR scheme [5] to generate initial partitions that identify clusters in the given circuit, which in turn promotes faster rate of convergence. Our iterative improvement-based flat multiway partitioner K-PM/LR improves K-FM by a surprising average margin of up to 86.2% and outperforms its counterpart recursive FM by up to 17.3% when tested on MCNC and large scale ISPD98 benchmark circuits [1].

The remainder of the paper is organized as follows; Section 2 presents the formulation of multiway partitioning problem. Section 3 provides the analysis of limitation conventional K-FM retains. Section 4 presents PM-based partitioning. Section 5 provides experimental results. Section 6 concludes the paper with our ongoing research.

2 Multiway Partitioning Formulation

Given a netlist to be partitioned into K non-empty disjoint blocks, we use $C = \{c_1, c_2, \cdots, c_p\}$, $N = \{n_1, n_2, \cdots, n_q\}$, and $B = \{b_1, b_2, \cdots, b_K\}$ to denote the set of cells, nets, and blocks, respectively. Each cell $c \in C$ may have non-uniform area. Under "cost 1" metric, a net has a cost of 1 if it spans more than 1 block, and 0 otherwise. Under "cost $k-1$" metric, a net has a cost of $k-1$ if it spans k blocks. A net with non-zero cost is called *cut*, and the sum of the cost of all cut nets is called *cutsize*. Then, the *optimal area-balanced K-way partitioning solution* of a given netlist satisfies the following conditions;

- Each cell is assigned to exactly one block;
- The total area of the cells in each block is within the following bounds;

$$(1-s) \cdot \frac{A}{K} \leq a_i \leq (1+s) \cdot \frac{A}{K}$$

  where A is the total area of all the cells, a_i is the total area of all the cells in block b_i $(1 \leq i \leq K)$, and s is a user-specified parameter controlling the allowable slack in the area constraint $(0 < s < 1)$;
- The cutsize is minimized.

Let $\gamma(n, b_i)$ denote the number of free cells of net n which are in block b_i. If there is at least one locked cell of n in b_i, $\gamma(n, b_i)$ becomes ∞. Then, let $\gamma'(n, b_i)$ denote sum of all $\gamma(n, b_k)$, for $1 \leq k \leq K$, $i \neq k$. $\gamma(n, b_i)$ and $\gamma'(n, b_i)$ measure how tightly net n is bound to block b_i, and all other blocks except for b_i, respectively. If n_c represents nets that are incident to c, the gain associated with moving cell c from block i to j based on cost 1 metric is;

$$\begin{aligned} g_c^1(i,j) &= |\{n \in n_c | \gamma'(n, b_j) = 1 \text{ and } \gamma(n, b_j) > 0\}| - \\ &\quad |\{n \in n_c | \gamma'(n, b_i) = 0 \text{ and } \gamma(n, b_i) > 1\}| \end{aligned}$$

The gain associated with moving cell c from block i to j based on cost $k-1$ metric is;

$$\begin{aligned} g_c^k(i,j) &= |\{n \in n_c | \gamma(n, b_i) = 1 \text{ and } \gamma(n, b_j) > 0\}| - \\ &\quad |\{n \in n_c | \gamma(n, b_j) = 0 \text{ and } \gamma(n, b_i) > 1\}| \end{aligned}$$

3 Limitation of Existing Approaches

Recursive approach is a simple extension of bipartitioning to multiway partitioning. It applies bipartitioning recursively until the desired number of partitions is obtained. It is computationally simple and fast, and many of the heuristics devised for bipartitioning can be applied to further reduce the current level cutsize. However, we note three major limitations of the recursive approach. First, cells can only move across the current level cutline, promoting local change in the current configuration as depicted in Figure 1-(a). The objective of recursive bipartitioning is to reduce the number of nets crossing the current level cutline in the absence of global information, which can trap the partitioner into a local minima and limit the solution quality. Second, recursive multiway partitioner can only minimize cost $k-1$ metric, not cost 1 metric. Third, it becomes harder and harder to reduce the cutsize as the bipartitioner performs deeper level cuts. Highly optimized 1st and 2nd level cuts can cause 3rd and 4th level cuts to cut through very dense clusters. Thus, this conflicting objective can cause recursive approach to produce low quality multiway partitioning solutions.

Sanchis [10] showed that the flat multiway partitioning approach obtained better quality solution compared to the recursive approach for small scale randomly generated circuits. In her K-way generalization of cell move based 2-way FM heuristic (often referred to as K-FM), $K(K-1)$ bucket structure are used to maintain cell gains. In practice, however, recursive FM is more widely used to generate multiway partitioning solution due to the empirical observation that K-FM is very susceptible of being trapped into a local minima.

We observe two major problems related to conventional K-FM. First, due to the high degree of flexibility, K-FM is prone to make wrong decision while dealing with many number of candidate cells and directions to move. This obviously increases the probability of getting stuck in the local minima in the absence of effective hill-climbing scheme. Table 1 shows the comparison of recursive FM (R-FM) and K-FM based on 4-way, 8-way, and 16-way partitioning result of MCNC and ISPD98 benchmark circuits [1] measured under cost 1 and cost $k-1$ metric.[1] K-FM that minimizes cost 1 is based on formulation given in [10], and K-FM for minimizing cost $k-1$ is based on [11]. As previously mentioned, R-FM minimizes only cost $k-1$, but we evaluate its multiway partitioning result with both metrics. As one can see, 50 runs of K-FM performs very poorly (almost 500% worse) compared to 20 runs of R-FM (we provide more detailed comparison in [6]).

Second problem is related to memory requirement. Each cell is associated with $K-1$ gain values, and each block has to maintain $K-1$ buckets for conventional K-FM algorithm. This translates into $O(N \cdot K(K-1))$ space complexity, where N denotes total number of cells. In case of large K or N, it requires prohibitively large amount of memory, causing K-FM to be undesirable in solving multiway partitioning problem for today's large scale circuits.

4 PM-based Multiway Partitioning

We propose a simple yet effective hill-climbing method called PM (Pairwise cell Movement) to enhance K-FM, which is done by reducing the multiway partitioning problem to sets of concurrent bipartitioning problems. PM passes are shown

[1] We note K-FM result on standard benchmark circuits is not well documented in the literature unlike recursive FM and its variants.

		cost 1						cost k-1					
circuits		4-way		8-way		16-way		4-way		8-way		16-way	
name	# cell	R-FM	K-FM	R-FM	K-FM	R-FM	K-FM	R-FM	K-FM	R-FM	K-FM	R-FM	K-FM
biomd	6514	157	630	223	777	300	945	187	776	339	776	558	776
s13207	8772	141	625	209	827	270	945	163	688	253	688	348	688
s15850	10470	161	811	228	1026	320	1167	175	901	254	901	375	901
s35932	18148	231	1818	294	2651	373	3375	274	1902	411	1902	580	1902
s38584	20995	213	2209	314	3194	434	3825	224	2704	350	2704	533	2704
avq.sm	21918	528	3007	767	4071	1008	5043	572	3128	894	3128	1257	3128
s38417	23949	280	1899	449	2506	604	2837	296	2197	494	2197	689	2197
avq.lg	25178	717	3324	1042	4252	1251	5308	769	3379	1178	3379	1502	3379
ibm01	12752	576	3212	857	4234	1462	4604	581	2690	904	4059	1615	4870
ibm02	19601	688	5984	2069	7138	3727	7725	740	1274	2215	3636	4356	7099
ibm03	23136	2596	6737	3512	8263	4454	8997	2811	6141	4216	9131	5765	10960
ibm04	27507	2290	8332	3751	10347	5154	11162	2369	7025	4039	10518	5962	13143
ibm05	29347	4225	8537	5760	9387	7310	9488	4751	8264	7183	13036	9746	18145
ibm06	32498	2096	8664	2954	10923	3914	12026	2343	6480	3668	9066	5466	12887
ibm07	45926	3069	12724	4375	15725	5955	16765	3193	9754	4853	14650	6836	20048
ibm08	51309	2945	12845	4532	16056	6031	17472	3040	11693	5048	15737	7043	21388
ibm09	53395	2838	15888	4759	19619	6327	21509	2918	14679	5104	16890	7131	23890
ibm10	69429	3163	20820	4888	26170	7047	27681	3319	16425	5173	21540	7716	29387
ibm11	70558	4685	21448	6059	27479	8168	29598	4799	18988	6360	25327	8855	35055
ibm12	71076	5258	23081	7946	28764	10776	29074	5387	18697	8407	26548	11644	37215
ibm13	84199	3102	24758	4390	30975	7258	31965	3278	20245	4803	30565	8063	41234
ibm14	147605	6451	38767	8424	49334	12038	51738	6842	29983	9251	46698	13456	61250
ibm15	161570	8310	48130	11465	64235	13966	67543	8701	41495	12689	53406	16106	76367
ibm16	183484	6228	54578	10372	65553	16205	68765	6303	48174	10864	60343	17452	79736
ibm17	185495	9326	64340	14733	75432	21857	80645	9494	53886	15336	63899	23591	89736
ibm18	210613	3952	53128	6588	65361	10327	68424	4070	43875	6902	56865	11208	79162
SUM	-	74226	446296	110960	554299	156536	588626	77599	375443	121188	497589	177853	677242
TIME	-	20.3	29.3	29.0	36.5	36.7	51.3	20.3	30.0	29.0	37.8	36.7	53.2

Table 1: Multiway partitioning of MCNC and ISPD98 benchmark circuits with recursive FM (R-FM) and flat K-way FM (K-FM) measured under cost 1 and cost $k-1$ metric. TIME reports total elapsed CPU hour for 20 (R-FM) and 50 (K-FM) runs of all 26 circuits.

to be effective in distributing clusters evenly into the multiple blocks to minimize the connections across the multiway cutlines. PM overcomes the limitation of conventional K-FM and provides partitioners the capability to explore a wider range of solution space effectively while ensuring convergence to satisfying suboptimal solutions.

4.1 Pairwise Cell Movement

In our Pairwise cell Movement (PM) approach, bipartitioning is applied to *pairs* of blocks so as to improve the quality of overall multiway partitioning. Cell moves are limited between paired blocks in this case, but PM can also promote global cell moves during subsequent passes that employ different pairing configurations. Starting with an initial K-way partition of the netlist, we pair all K blocks before a pass of PM begins. Then, we initialize and update gain of each cell moving from block b_i to b_j only if they are paired. Note that this is a restricted version of K-FM, where some of the cell move directions are ignored.

Now each cell is associated with single gain value, and each block maintains single bucket. This translates into a lower $O(N \cdot K)$ space requirement compared to the conventional K-FM. However, PM only considers $O(K)$ directions out of $O(K(K-1))$, which ignores many directions with even higher gains. Then, a natural question arises if PM will adversely affect the partitioner with this restriction. Figure 2 shows a typical behavior of K-FM and PM with random initial partition. K-FM converges to a local minima quickly, while PM searches for better solution with a slight increase of runtime. A possible explanation is that during each PM pass, the partitioner focuses on removing clusters that straddle the cutline between paired blocks. Then the subsequent PM passes are used to redistribute clusters evenly into the K blocks to minimize the connections across the multiway cutlines. Note that it is possible to end up with negative gain between some non-paired blocks at the end of a PM pass. K-FM does not allow this case; a positive overall pass gain always means cutsize improvement with respect to all possible pairs. Our stopping criteria of PM-based run is based on the *overall* gain computed from the gain between all possible pairs of blocks to ensure the convergence. This is how PM provides the partitioner with hill-climbing capability that overcomes the drawback of K-FM.

4.2 Block Pairing and Initial Partitioning

One important decision to make at the end of each PM pass is how to come up with pairing configuration for the next PM pass. PM requires a matching-like pairing of blocks as shown in Figure 3-(a), and we observe the following possible strategies in regards to the selection of $K/2$ pairs out of $K(K-1)/2$;

- *random* : randomly pair blocks. It serves as a reference point to other strategies.
- *exhaustive* : rotate among all possible pairing configurations, as shown in Figure 3-(b). The purpose is to apply the same number of PM pass to each possible pairing configuration, giving the partitioner a chance

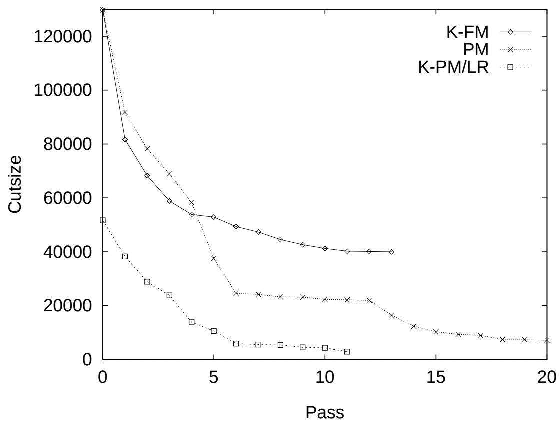

Figure 2: Typical cutsize reduction trend of K-FM, PM, and K-PM/LR for 8-way partitioning of ibm18

to move cells between any blocks during subsequent PM passes.

- *cut-based* : always pair two most tightly or loosely connected blocks, measure in terms of cutsize.
- *gain-based* : always pair two blocks between which the cutsize reduction is maximum or minimum during last p PM passes.

We empirically observe from related experiment [6] that gain-based pairing that selects two blocks between which the cutsize reduction is maximum during the last PM pass produces the best result. The *gain-of-pair* is computed by comparing the number of nets that span both blocks before and after a pass. Then, we use a heap of size $K(K-1)/2$ that sorts pairs in descending order of their gains from the last pass. Another decision to be made is when to terminate the current run. In *immediate* stopping, the partitioner stops right after it encounters the first non-positive pass gain. In *exhaustive* stopping, the partitioner stops if all (or some portion of) possible pairing configurations consecutively can't improve the partition. We observe from related experiment [6] that immediate stopping produces almost the same quality solution within only a fraction of runtime compared to exhaustive stopping.

Another important issue to be addressed is on initial partition. Figure 2 shows the typical behavior of PM (based on random initial partition) and K-PM/LR (based on initial partition by recursively applying the existing LR scheme [5]). LR is a simple yet effective approach to dynamically identify and remove clusters that straddle the cutline in bipartitioning. We use limited number of LR passes (usually less than 3) at each recursive level cut for the generation of K blocks, which requires little CPU time in most cases. However, the impact on the performance of PM is noticeable. The rate of convergence is faster compared to random initial partition, and PM obtains better quality solutions as revealed in Figure 2. Our most enhanced multiway partitioner named K-PM/LR combines recursive LR-based initial partitioning and PM passes as shown in Table 2.

5 Experimental Result

We have implemented our K-PM/LR algorithm that combines LR [5] and PM, compiled with gcc v2.4, and tested on

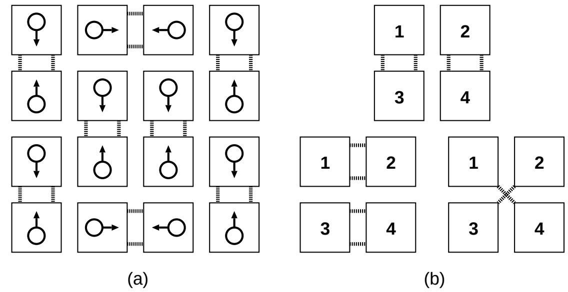

Figure 3: (a) One possible block pairing configuration for 16-way partitioning, (b) All 3 possible block pairing configurations for 4-way partitioning

```
K-PM/LR()
for (r = 1 to total_run)
    LR_INIT_PARTITION();
    while (¬ IMMEDIATE_STOP())
        GAIN_BASED_BLOCK_PAIRING();
        PM_PASS();
```

Table 2: Description of K-PM/LR

SUN ULTRA SPARC-1, 143Mhz. The benchmark circuits are from MCNC and ISPD98 [1] suits. The area of the cells is uniform, and all pads are included to be partitioned. The recursive bipartitioning algorithms use [0.45, 0.55] balancing constraint for all level cuts, whereas 4-way, 8-way, and 16-way flat algorithms use [$0.45^2 = 0.203$, $0.55^2 = 0.303$], [$0.45^3 = 0.091$, $0.55^3 = 0.166$], and [$0.45^4 = 0.041$, $0.55^4 = 0.092$]. All cutsizes are based on 20 runs, and runtimes are measured in hours. We report the sum of total elapsed CPU time of each algorithm.

Table 3 shows the comparison of recursive FM (R-FM) and K-PM/LR based on 4-way, 8-way, and 16-way partitioning result measured under cost 1 and cost $k-1$ metric. As one can see, K-PM/LR significantly improves K-FM by up to 86.2% based on the comparison with Table 1, and outperforms R-FM by up to 17.3%. In addition, cost $k-1$ results by K-PM/LR is so highly optimized that it almost matches cost 1 cutsize. In other words, most of the cut nets span only 2 blocks. Our flat multiway partitioner K-PM/LR also obtains comparable result (within 5%) when compared to the state-of-the-art recursive bipartitioner hMetis (recent result available at `http://vlsicad.cs.ucla.edu/~cheese`). Note that hMetis uses hierarchical clustering during partitioning. We expect that our result will further improve when combined with proper clustering schemes. Our technical report [6] provides more detailed experimental result.

6 Conclusion & Ongoing Work

We proposed a simple yet effective method to improve the iterative improvement-based multiway partitioner by reducing the multiway partitioning problem to sets of concurrent bipartitioning problems. The main contribution of our study is first to reveal the poor performance of conventional improvement-based multiway partitioner K-FM and next to provide detailed analysis as well as effective way to overcome the drawback of K-FM. The result is an effective and efficient

		cost 1						cost k-1					
circuits		4-way		8-way		16-way		4-way		8-way		16-way	
name	# cell	R-FM	K-PM	R-FM	K-PM	R-FM	K-PM	R-FM	K-PM	R-FM	K-PM	R-FM	K-PM
biomd	6514	157	172	223	261	300	374	187	198	339	394	558	602
s13207	8772	141	171	209	219	270	368	163	204	253	331	348	490
s15850	10470	161	140	228	264	320	412	175	165	254	286	375	486
s35932	18148	231	154	294	291	373	352	274	175	411	379	580	440
s38584	20995	213	191	314	368	434	600	224	228	350	459	533	669
avq.sm	21918	528	562	767	794	1008	1100	572	587	894	828	1257	1159
s38417	23949	280	168	449	220	604	441	296	170	494	301	689	535
avq.lg	25178	717	625	1042	927	1251	1102	769	625	1178	1065	1502	1400
ibm01	12752	576	479	857	1020	1462	1699	581	542	904	1109	1615	1821
ibm02	19601	688	639	2069	1751	3727	3592	740	662	2215	1892	4356	4152
ibm03	23136	2596	2537	3512	3882	4454	5736	2811	2530	4216	4119	5765	5662
ibm04	27507	2290	2192	3751	3559	5154	5349	2369	2201	4039	3671	5962	5766
ibm05	29347	4225	3442	5760	4834	7310	6419	4751	3692	7183	6543	9746	9344
ibm06	32498	2096	1944	2954	3198	3914	4815	2343	2245	3668	3988	5466	5900
ibm07	45926	3069	2843	4375	4398	5955	6854	3193	2949	4853	4707	6836	6854
ibm08	51309	2945	3161	4532	4466	6031	6477	3040	3529	5048	5426	7043	7364
ibm09	53395	2838	2555	4759	4115	6327	6046	2918	2965	5104	4187	7131	5978
ibm10	69429	3163	2996	4888	5252	7047	8559	3319	3229	5173	5518	7716	8525
ibm11	70558	4685	3189	6059	6086	8168	8871	4799	3646	6360	5321	8855	8420
ibm12	71076	5258	4429	7946	7736	10776	11000	5387	4615	8407	7530	11644	10495
ibm13	84199	3102	2325	4390	3570	7258	7066	3278	2374	4803	3667	8063	7382
ibm14	147605	6451	5233	8424	6753	12038	9854	6842	5098	9251	7427	13456	12476
ibm15	161570	8310	6344	11465	8965	13966	11345	8701	8049	12689	11008	16106	14448
ibm16	183484	6228	5034	10372	7543	16205	10456	6303	5992	10864	9322	17452	14901
ibm17	185495	9326	6738	14733	10654	21857	17653	9494	6779	15336	11818	23591	20830
ibm18	210613	3952	3123	6588	5765	10327	9653	4070	3814	6902	6982	11208	11692
SUM	-	74226	61386	110960	96891	156536	146193	77599	67263	121188	108278	177853	167791
TIME	-	20.3	19.4	29.0	29.4	36.7	37.4	20.3	19.3	29.0	28.4	36.7	35.1

Table 3: Multiway partitioning of MCNC and ISPD98 benchmark circuits with R-FM (Recursive FM) and K-PM/LR measured under cost 1 and cost $k-1$ metric. TIME reports total elapsed CPU hour for 20 runs of all 26 circuits.

algorithm K-PM/LR that improves conventional flat multiway partitioning algorithm significantly and outperforms recursive algorithms. Our ongoing study includes; (i) adaptation of clustering to further reduce cutsize and runtime, (ii) application of K-PM/LR to quadrisection based placement.

Acknowledgement

We thank Charles Alpert at IBM Austin Research Laboratory, George Karypis at University of Minnesota, Laura Sanchis at Colgate University, and Narayanan Shivakumar at Stanford University for sharing their experience on multiway partitioning. We would also like to thank Honching Li at UCLA for his helpful comments regarding this paper.

References

[1] C. J. Alpert. The ISPD98 circuit benchmark suite. In *Proc. Int. Symp. on Physical Design*, pages 80–85, 1998.

[2] C. J. Alpert and A. B. Kahng. Multi-way partitioning via spacefilling curves and dynamic programming. In *Proc. Design Automation Conf.*, pages 652–657, 1994.

[3] P. Chan, M. Schlag, and J. Zien. Spectral k-way ratio-cut partitioning and clustering. In *Proc. Design Automation Conf.*, pages 749–754, 1993.

[4] J. Cong, W. Labio, and N. Shivakumar. Multi-way VLSI circuit partitioning based on dual net representation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 396–409, 1996.

[5] J. Cong, H. P. Li, S. K. Lim, T. Shibuya, and D. Xu. Large scale circuit partitioning with loose/stable net removal and signal flow based clustering. In *Proc. Int. Conf. on Computer-Aided Design*, pages 441–446, 1997.

[6] J. Cong and S. K. Lim. Multiway partitioning with pairwise movement. Technical Report 980029, CS Dept. of UCLA, Aug. 1998.

[7] C. Fiduccia and R. Mattheyses. A linear time heuristic for improving network partitions. In *Proc. Design Automation Conf.*, pages 175–181, 1982.

[8] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Technical Report 95-064, CS Dept. of Univ. of Minnesota, 1995.

[9] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning of electrical circuits. *Bell System Technical Journal*, pages 291–307, 1970.

[10] L. A. Sanchis. Multiple-way network partitioning. *IEEE Trans. on Computers*, pages 62–81, 1989.

[11] L. A. Sanchis. Multiple-way network partitioning with different cost functions. *IEEE Trans. on Computers*, pages 1500–1504, 1993.

[12] C. W. Yeh, C. K. Cheng, and T. T. Lin. A general purpose multiple-way partitioning algorithm. In *Proc. Design Automation Conf.*, pages 421–426, 1991.